A Machine Learning Approach for Estimating Overtime Allocation in Software Development Projects

Hammed Adeleye Mojeed
Department of Computer System Architecture, Faculty of Electronics, Telecommunication and Informatics, Gdansk University of Technology
Gdansk, Poland *hammed.mojeed@pg.edu.pl*

Rafal Szlapczynski
Department of Applied Computer Science, Institute of Ocean Engineering and Ship Technology, Gdansk University of Technology
Gdansk, Poland *rafal.szlapczynski@pg.edu.pl*

Abstract

Overtime planning in software projects has traditionally been approached with search-based multi-objective optimization algorithms. However, the explicit solutions produced by these algorithms often lack applicability and acceptance in the software industry due to their disregard for project managers' intuitive knowledge. This study presents a machine learning model that learns the preferred overtime allocation patterns from solutions annotated by project managers and applied to four publicly available software development projects. The model was trained using 1092 instances of annotated solutions gathered from software houses, and the Random Forest Regression (RFR) algorithm was used to estimate the PMs' preference. The evaluation results using MAE, RMSE, and R2 revealed that RFR exhibits excellent predictive power in this domain with minimal error. RFR also outperformed the baseline regression models in all the performance measures. The proposed machine learning approach provides a reliable and effective tool for estimating project managers' preferences for overtime plans.

Keywords: Software Overtime Planning, Software Project Planning, Machine Learning, Random Forest Regression.

1. Introduction

Planning a software project is a complex and highly dynamic task characterized by uncertainties and the risk of overrun in duration and cost. Although PMs are provided with automated project planning tools, software development teams still suffer from unplanned overtime as the only option when the project encounters "mission creep" or an excessive change in requirements [4]. This issue of unplanned overtime has been a persistent challenge in the software industry, leading to negative impacts on developers [6], [14] and the quality of the software they build [3]. These findings have drawn researchers' attention to more proactive overtime planning, which is the focus of our research.

The current approach modeled software overtime allocation as a multi-objective optimization problem considering its effects on project duration, cost, overrun risk, and quality [2, 3], [8], leading to a new field called Software Overtime Planning (SOP). The first search-based optimization formulation of SOP was introduced by Ferrucci et al. [3]. Subsequent studies have extended their work with multi-objective evolutionary algorithms [2], [12] and multi-objective memetic algorithms [8] to produce optimal overtime plans. Ferruci et al. [3] applied NSGA-IIv, a variant of NSGA-II specifically designed for overtime scheduling, to find the optimal overtime allocations. Similarly, Sarro et al. [12] applied Adaptivevsc, a variant of NSGA-II that efficiently combines the crossover operator used in [3] and adaptive genetic operators to produce a dynamic strategy for selecting genetic operators as optimization progresses. Using the same NSGA-II, De Barros and De Araujo [2] incorporated the already-established effect of overtime on software quality into

SOP formulation by simulating the defects introduced by developers during overtime as they affect project cost and duration. Mojeed et al. [8] introduced a memetic approach to SOP using the same experimental setting and datasets as in [2] and a multi-objective shuffled frog-leaping algorithm (MOSFLA) as the search method.

These existing studies in SOP have produced quality solutions for Project Managers (PMs) to allocate overtime better. However, these explicit solutions were generated without the input of PMs, which has affected their acceptance by the PMs. Studies revealed that PMs favor their intuitive judgments for the initial overtime allocation solutions[9], [12]. To address this problem, this study presents a machine-learning approach to solving the SOP problem by building an estimation model that learns from real-world PM annotations. To our knowledge, this is the first time a machine-learning approach has been applied to SOP. The specific contributions of this work are collecting annotated software overtime planning solutions data by PMs from the industry and developing a Random Forest Regression (RFR) model based on the collected data to estimate the PM's preference for overtime allocation.

2. Methodology

Building a machine learning model for software development overtime planning requires project managers to annotate real-world software project schedule datasets. To do this, six real-world software project data collected by [2] were obtained from https://github.com/luizaraujojr/GECCO2016. ACAD manages university students in a portal, including registrations, classes, and teacher records. PSOA manages users' authorization and authentication from enterprise systems. WEBAMHS controls the air traffic routing system for airlines. WEBMET manages meteorological information in a database. The dataset (provided in XML) specifies the project's tasks, function points (FP) sizes, and dependencies, as presented in Table 1.

Table 1. Properties of the obtained Software Projects Datasets

Project	No of Activities	No of Dependencies	Function points
ACAD	40	39	185
WEBMET	44	33	225
WEBAMHS	60	45	381
PSOA	72	84	290

The Work Packages (WP) based features of the software projects were extracted. Given the size of a WP in FP, its expected duration, described in Equation 1, is estimated using a mean productivity value of 27.8 FP/developer-month as recommended by [5] for IT projects and previously used in existing studies in software overtime planning [2].

$$D_{duration(wp_i)} = 30\left(\frac{effort(wp_i)}{27.8}\right) \quad (1)$$

Since a project schedule has to be built before producing overtime allocation plans for the software projects, the widely used Critical Path (CP) method [2, 3, 8, 12] was adopted to build schedules for the projects and estimate the shortest possible duration to complete them. The extracted information based on WPs in the project data is shown in Table 2.

Table 2. Extracted Properties of the pre-processed Software Projects Dataset

Project	No of WPs	No of WP`s Dependencies	Function points	Shortest Possible Duration (days)
ACAD	10	9	185	181
WEBMET	11	17	225	243
WEBAMHS	15	20	381	413
PSOA	18	34	290	316

2.1. Overtime Allocation

An overtime plan modeled as a finite set of N (the number of WPs in the project) integer values in the interval [0, 4], representing the number of daily overtime hours assigned to each *WP,* is adopted in this work. The overtime plan allocation is evaluated based on the three objectives: total overtime hours, cost due to overtime, and code quality. Their mathematical definition, as adopted, can be found in our previous work [9].

For each project, several overtime planning solutions satisfying the defined objectives were generated using the multi-objective random search from the JMetal MOEA framework library. The random search was used to explore the search space more extensively and avoid exploiting a particular region of the space since our goal is to generate as many feasible solutions as possible to train a machine learning model. Generated solutions were constrained to accommodate the three well-known overtime management strategies: Critical Path Management (CPM), Margarine (MAR), and Second Half (SH).

2.2. Solution Annotation and ML Model Building

Since the ML approach requires a labeled dataset to train models, the generated overtime allocation solutions were prepared in CSV format and presented to PMs along with the extracted characteristics of the software projects for annotation. 20 software project managers with experience in overtime allocation were recruited to evaluate the overtime plans based on their expertise. The project managers give numeric evaluation scores between 1 and 100 to each solution instance, adapted from [13] to estimate their preference.

Table 3. Description of the final datasets from the software projects

Project	instances (solutions)	Features (Tasks)	% MAR	%CPM	%SH	%Random
ACAD	282	10	68	71	73	70
WEBMET	269	11	63	68	71	67
WEBAMHS	275	15	69	70	72	64
PSOA	266	18	65	63	70	68

The annotations received were further processed to remove outliers by computing the deviation of the values from the mean. This step is necessary to remove the bias in the estimation as the PM evaluations are characterized mainly by subjective judgment. The final score is the mean of the closest annotation values representing experts' majority agreement. From the four software projects obtained, 1092 cleaned annotated solutions were produced, as described in Table 3.

Random Forest Regression (RFR), a supervised learning model that uses an ensemble learning method to build a regression model[1, 15], was used in this work. RFR creates several decision trees (DT) during training time and delivers the output of the average prediction of all the DTs. RFR was chosen for the study due to its ability to handle non-linear relationships in data effectively. Moreover, RFR has been successfully applied to solve software engineering problems such as effort estimation[10]. The parameter settings n_estimator=150, min_samples_split=0.03, max_features=Log_2(n-features), and max_depth=10 were used following the recommendation of Probst [11]. Its performance on the dataset was evaluated using MAE, RMSE, and R^2.

3. Results and Discussion

The RFR model was implemented using the WEKA classifier library in Java. The dataset for each project is split into 80% training and 20% test set. The algorithm was trained and validated using the 10-fold cross-validation approach. It has been proven experimentally to produce minimal and unbiased test error rates with low variance [7]. Performance evaluation results of RFR on the annotated dataset are presented in Table 4.

Concerning MAE, RFR produced good results with minimal average error. The best result was recorded in the WEBMET project with an average error of 7.85, and the least

results were obtained in WEBAMHS with an average error of 10.40. Regarding RMSE, a similar pattern of results was obtained, with the best performance recorded in the WEBMET project and the least in WEBAMHS. Considering the R^2 scores, RFR performed considerably well in estimating the PM's satisfaction with the overtime allocation, producing a maximum value of 0.89 in the WEBMET project. The RFR model can predict PM satisfaction with an average of 0.82% accuracy.

Table 4. RFR Performance Results

Project	MAE	RMSE	R2
ACAD	8.81	12.49	0.84
WEBMET	7.85	11.56	0.89
WEBAMHS	10.40	13.12	0.78
PSOA	9.64	12.86	0.80
Average	**9.18**	**12.51**	**0.82**

Concerning the model's scalability, it was observed that as the project size (measured in WP size and FP) increases, the RFR performance decreases, as indicated by the trendline in Figures 2 and 3. This might be due to the existence of tasks with high developer efforts in large projects. These kinds of tasks are difficult to allocate overtime and contribute heavily to the final overtime of the project. Also, observing the shape of the curves, it is clear that FP models the relationship between RFR performance and project size better than WP.

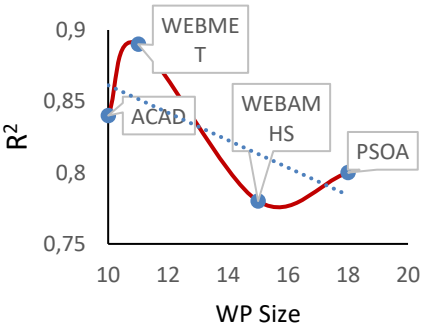

Fig. 2. Performance of RFR based on the WP size

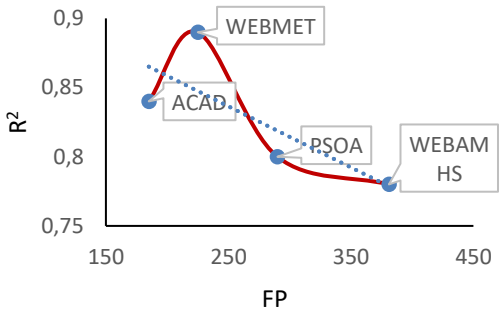

Fig. 3. R^2 Performance of RFR based on FP

Table 5 compares the model's performance with SLR, MLR, and SVR. It can be observed in Table 5 that the proposed RFR outperformed all the baseline regression models in RMSE and R^2 across all the projects with an average improvement of 11.99% and 30.04% in performance, respectively. In MAE, RFR gave better results in three of the projects except in ACAD, where SVR slightly outperformed RFR. These results indicated the effectiveness of the RFR model in estimating PM preference.

Table 5. Comparison between RFR and baseline Regression models

Projects	*MAE*				*RMSE*				R^2			
	RFR	SLR	MLR	SVR	RFR	SLR	MLR	SVR	RFR	SLR	MLR	SVR
ACAD	8.91	10.62	9.54	**8.24**	**12.49**	15.04	13.56	13.00	**0.84**	0.58	0.65	0.71
WEBMET	**7.85**	11.89	8.26	9.43	**11.56**	14.25	12.07	12.56	**0.89**	0.60	0.69	0.74
WEBAMHS	**10.26**	12.63	11.87	10.95	**13.12**	16.24	15.04	14.82	**0.78**	0.52	0.60	0.68
PSOA	**9.64**	11.45	10.45	10.08	**12.86**	15.98	14.24	13.98	**0.80**	0.55	0.63	0.70

With its promising results, the proposed model can be applied to software projects of similar or closely related characteristics. Estimating the effort and size of software projects using analogy is well-established in the literature and can be extended to overtime estimation. Overtime plans can be randomly generated for new and similar projects based

on the industry overtime allocation strategies. Then, each solution is tested using the built model to predict the PM's satisfaction with the solutions.

4. Conclusion

This study developed a machine learning model based on RFR that learns the overtime allocation patterns preferred by PMs in planning software projects. Performance evaluation results show the suitability and predictive effectiveness of RFR in estimating overtime plans for software development projects. The study also confirmed the superiority of RFR to the baseline regression models in estimation accuracy and error rate. Our approach has produced a simplified method for solving the SOP problem using machine learning.

References

1. Alsariera, Y.A., Balogun, A.O., Adeyemo, V.E., Tarawneh, O.H., Mojeed, H.A.: Intelligent Tree-Based Ensemble Approaches for Phishing Website Detection. J. Eng. Sci. Technol. 17 (1), 563–582 (2022)
2. DeO Barros, M., De Araujo, L.A.O.: Learning overtime dynamics through multiobjective optimization. In: GECCO 2016 - Proceedings of the 2016 Genetic and Evolutionary Computation Conference. pp. 1061–1068. ACM, Inc (2016)
3. Ferrucci, F., Harman, M., Ren, J., Sarro, F.: Not going to take this anymore: Multi-objective overtime planning for software engineering projects. In: 2013 35th International Conference on Software Engineering (ICSE). pp. 462–471. (2013)
4. Ferrucci, F., Harman, M., Sarro, F.: Search-Based Software Project Management. In: Ruhe, G. and Wohlin, C. (eds.) Software Project Management in a Changing World. pp. 373–399. Springer (2014)
5. Jones, C.: Software assessments, benchmarks, and best practices. Addison-Wesley Longman Publishing Co., Inc. (2000)
6. Kleppa, E., Sanne, B., Tell, G.S.: Working Overtime is Associated With Anxiety and Depression: The Hordaland Health Study. J. Occup. Environ. Med. 50 (6), 658–666 (2008)
7. Kuhn, M., Johnson, K., others: Applied predictive modeling. Springer (2013)
8. Mojeed, H.A., Bajeh, A.O., Balogun, A.O., Adeleke, H.O.: Memetic approach for multi-objective overtime planning in software engineering projects. J. Eng. Sci. Technol. 14 (6), 3213–3233 (2019)
9. Mojeed, H.A., Szlapczynski, R.: Machine Learning Assisted Interactive Multi-objectives Optimization Framework: A Proposed Formulation and Method for Overtime Planning in Software Development Projects. Lect. Notes Comput. Sci. 14125 LNAI 415–426 (2023)
10. Priya Varshini, A.G., Anitha Kumari, K., Varadarajan, V.: Estimating software development efforts using a random forest-based stacked ensemble approach. Electron. 10 (10), (2021)
11. Probst, P.: Hyperparameters, tuning and meta-learning for random forest and other machine learning algorithms. (2019)
12. Sarro, F., Ferrucci, F., Harman, M., Manna, A., Ren, J.: Adaptive multi-objective evolutionary algorithms for overtime planning in software projects. IEEE Trans. Softw. Eng. 43 (10), 898–917 (2017)
13. Simons, C.L., Smith, J., White, P.: Interactive ant colony optimization (iACO) for early lifecycle software design. Swarm Intell. 8 (2), 139–157 (2014)
14. Swenson, D.X.: A Systems Model of Overtime Effects on Software Development Team Performance. The College of St. Scholastica (2014)
15. Usman-Hamza, F.E., Balogun, A.O., Nasiru, S.K., Capretz, L.F., Mojeed, H.A., Salihu, S.A., Akintola, A.G., Mabayoje, M.A., Awotunde, J.B.: Empirical analysis of tree-based classification models for customer churn prediction. Sci. African. 23, e02054 (2024)